Prototyping Platform for Performance Evaluation of SHA-3 Candidates

Kazuyuki Kobayashi*, Jun Ikegami*, Miroslav Knežević†, Eric Xu Guo‡, Shin'ichiro Matsuo§,
Sinan Huang‡, Leyla Nazhandali‡, Ünal Kocabaş†, Junfeng Fan†, Akashi Satoh¶, Ingrid Verbauwhede†,
Kazuo Sakiyama* and Kazuo Ohta*

* The University of Electro-Communications, 1-5-1, Chofugaoka, Chofu, Tokyo 182-8585, Japan
Email: {k-kazu, jike, saki, ota}@ice.uec.ac.jp

† Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
Email: {miroslav.knezevic, Unal.Kocabas, Junfeng.Fan, ingrid.verbauwhede}@esat.kuleuven.be

‡ Electrical and Computer Engineering Dept. Virginia Polytechnic Institute and State University Blacksburg, VA 24061
Email: {xuguo, shuang86, leyla}@vt.edu

§ National Institute of Information and Communications Technology,
4-2-1 Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan
Email: smatsuo@nict.go.jp

¶ Research Center for Information Security, National Institute of Advanced Industrial Science and Technology,
1-18-13, Sotokanda, Chiyoda, Tokyo 101-0021, Japan
Email: akashi.satoh@aist.go.jp

***Abstract*—The objective of the SHA-3 NIST competition is to select, from multiple competing candidates, a standard algorithm for cryptographic hashing. The selected winner must have adequate cryptographic properties and good implementation characteristics over a wide range of target platforms, including both software and hardware. Performance evaluation in hardware is particularly challenging because of the large design space, wide range of target technologies, and multitude of optimization criteria. We describe the efforts of three research groups to evaluate SHA-3 candidates using a common prototyping platform. Using a SASEBO-GII FPGA board as a starting point, we evaluate the performance of the 14 remaining SHA-3 candidates with respect to area, throughput, and power consumption. Our approach defines a standard testing harness for SHA-3 candidates, including the interface specifications for the SHA-3 module on the SASEBO testing board.**

***Keywords*—hardware implementation, hardware evaluation, hash function, SHA-3**

I. Introduction

We briefly explain the origin of the SHA-3 NIST competition, and the need for a standard hardware evaluation mechanism.

A. Background

Since collisions on standard hash functions were reported in 2004, improvements to hash attack methods and hash algorithms have progressed at a similar, rapid pace [1]. For this reason, NIST decided to initiate development of a new hash standard.

The selection of winning candidates is driven by considering security as well as implementation efficiency of the proposed hash algorithms in hardware and software. However, systematic cryptanalysis of hash functions is not well established, and it is hard to measure the cryptographic strength of a hash function beyond obvious metrics such as digest length. For this reason the implementation efficiency of hardware and software plays a vital role in selecting the finalists.

Several ongoing projects are evaluating the hardware efficiency of the SHA-3 candidates [3], [4], [5], [6], but the validity and consistency of the evaluation criteria and methods of such research are not yet well discussed. In order to evaluate the hardware efficiency over a set of SHA-3 candidates, we need to fix an evaluation environment (i.e. platform), implementation method (i.e. design strategy), and performance comparison method (i.e. evaluation criteria). Consensus on such points is required for a fair comparison.

Because there are many possible application scenarios for hash functions, a universal set of hardware evaluation criteria may be hard to define. In this paper, we therefore focus on criteria that are consistent and fair on a given platform.

B. Related Work

Recently, several research groups have proposed comprehensive performance evaluation methods, which evaluate a set of hash algorithms on a common platform. These proposals use a synthesis target.

- Tillich et al. developed an RTL Verilog code for all SHA-3 candidates. They present syntheses results on UMC 180nm technology [3].
- Gaj et al. developed a scripting system called ATHENA, targeted at FPGA [7]. A fair comparison is achieved by defining a standard interface and by automatic design space exploration.
- Baldwin et al. [8] propose several hash candidates, as well as a standard interface to achieve a fair comparison.

C. Our Contribution

In this paper, we propose a platform, design strategy, and evaluation criteria for a fair hardware evaluation of the SHA-3 candidates. In comparison with the above efforts, we use a prototyping approach rather than synthesis for ASIC. We map each of the hash candidates onto a SASEBO-GII FPGA board [9]. The candidates are then evaluated on throughput, latency, hardware cost, and power consumption. Our evaluation methods allow determination of hardware performance independently from the interface specifications. The source code of the SHA-3 candidates is written in RTL Verilog, and can be reused for future ASIC development. This ASIC will be tested on a similar SASEBO platform.

In the next sections, we detail the hardware evaluation platform, the design strategy used for SHA-3 candidates, and the proposed performance evaluation metrics.

II. Hardware Evaluation Platform for SHA-3 Candidates

A. Overview

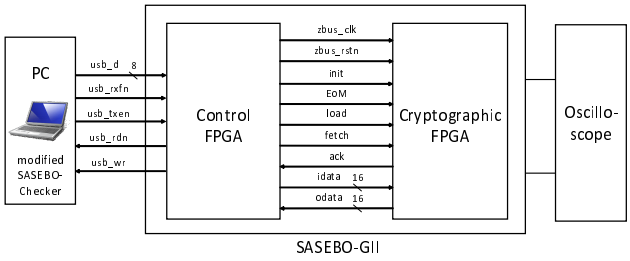

Fig. 1. Evaluation Environment Using SASEBO-GII.

Figure 1 illustrates the target platform for our evaluation, which includes a SASEBO-GII board and a PC. The SASEBO board contains two FPGAs: a control FPGA and cryptographic FPGA. On the PC, a test program enables a user to enter a sample message, which is transmitted to the control FPGA via USB. The control FPGA controls the data flow to send this message to the cryptographic FPGA, where hash operations are performed. After the hash operations, the digest is returned to the PC through the control FPGA. As illustrated in Fig. 1, the interface between the control FPGA and cryptographic FPGA is fixed and common among all SHA-3 candidates.

The control FPGA checks the latency of a single hash operation for input data performed in the cryptographic FPGA and reports the number of clock cycles to the PC. The PC then reports two different performance metrics: the number of clock cycles including the cycles for receiving input data and that excluding the cycles for the data input.

During hashing of a message, we also measure the power of the hashing operation. This trace, in combination with performance data, enables a precise characterization of the power dissipation and energy consumption of the SHA-3 candidate on the cryptographic FPGA.

B. Hardware Interface

A key concept in our approach is the use of a standard interface to integrate hash algorithms inside the cryptographic FPGA. In this section, we describe the major principles of this interface. We also compare our ideas with those of several other proposals. A hardware interface for a SHA-3 module emulates similar functionality as the software API interface. The hardware interface therefore needs to address the following issues.

Handshake protocol: The hash interface needs to synchronize data transfer between the SHA-3 module and the environment. This is done using a handshake protocol and a *master* protocol can be distinguished from a *slave* protocol depending on which party takes the initiative to establish synchronization. The interface in our proposal uses a slave protocol.

Word length: Typical block and digest lengths are wider (e.g. 512 bits) than the word length that can be provided by standard platforms (e.g. 32 bits), so that each hash operation will result in several data transfers. While this overhead is typically ignored by hardware designers, it is inherently part of the integration effort of the SHA-3 module. All of the interface proposals leave the standard interface word length undefined, though they implicitly assume 32 bits. In our proposal, we use a 16-bit interface, which is defined by the data-bus between the control FPGA and cryptographic FPGA.

Control: The functions of the software API need to be translated to equivalent hardware control signals. Our approach is to define additional control signals on the interface, for example to indicate message start and end.

Padding: Finally, padding may or may not be included in the SHA-3 hardware module. In the latter case, the hardware module implicitly assumes that an integral number of blocks will be provided for each digest. Common padding schemes are defined by in-band data formatting, and this makes it possible to implement padding outside of the hardware module. Our interface proposals leave padding to the SHA-3 designer. We assume hardware padding will only be implemented at the word level.

Note that there are many possible solutions to the interface problem, and we present one possible approach. We observe that the key issue for a fair comparison is to use a *common* interface for all candidates. In addition, we will show that our performance evaluation mechanism allows factoring out of the overhead of the interface communication.

III. Design Strategy

Besides a standard platform, our approach also defines a design strategy.

The previous work for evaluating hardware performance has been executed without using a standardized architecture, i.e. different architectures are used.

In this paper, we evaluate the performance of SHA-3 candidates when they are used in a real system. We also use performance metrics that enable us to separate the interface operation from the hash operation.

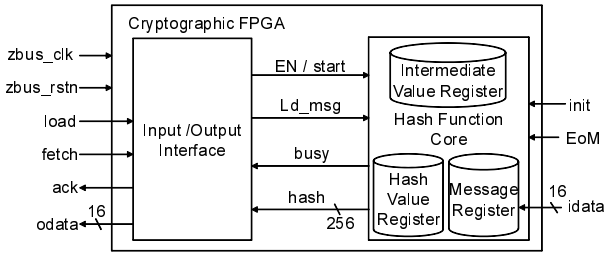

Fig. 2. Architecture of Cryptographic FPGA.

Figure 2 shows the detailed architecture of the cryptographic FPGA that we use for evaluating hardware performance. The cryptographic FPGA consists of an interface block that controls input and output, and a core function block that executes a hashing process. There are several SHA-3 candidates that need to keep an input message during the hashing process. In our environment, a message register can be prepared in the core function block. This architecture can reuse the message register for the next message block. Also, we may be able to prepare an output register in the core function block that keeps the result (e.g. 256-bit hash value). We used output register for all second-round candidates in this paper.

IV. Evaluation Criteria for Hardware SHA-3 Candidates

The third component of our evaluation strategy is the performance evaluation criteria. These are explained in this section.

A. Evaluation Items

We implement 14 SHA-3 hash candidates on the cryptographic FPGA, Xilinx Virtex-5 (xc5vlx30-3ff324) on SASEBO-GII. We check the hardware performance in terms of speed and hardware cost. The speed performance is evaluated with its latency or throughput that is calculated with the input block size, maximum clock frequency, and total number of clock cycles with or without communication overheads. The cost performance is evaluated by the number of slices, registers, and LUTs. A hash function that has high throughput with a low hardware cost is regarded as efficient. The power consumption of the hash design is measured during a complete hash operation. The energy cost is the integral of the power consumption over the period of hash operation. The energy cost is normalized to the message block length in order to obtain a standardized nJ/bit metric.

B. Speed Performance Metrics

We use the following notations to explain the criteria:

B : Input block size
I : Total number of clock cycles
D : Critical path delay
Th : Throughput
f_{max} : Maximum clock frequency
M : Size of message without padding
M_p : Size of message with padding

With an input block size, we use the result as the next input data to proceed with the entire hashing process. The clock cycles necessary for hashing M-bit data can be expressed as

$$I = \frac{M_p}{B}(I_{in} + I_{core}) + I_{final} + I_{out} \ . \quad (1)$$

Here, $\frac{M_p}{B}$ is the number of hash core operations when the hash core can perform B-bit data in one operation. I_{in}, I_{core}, I_{final}, and I_{out} denote the number of clock cycles used to input data, execute the hashing process in the core function block, perform the final calculation process and output the hash results, respectively. Note that the coefficients of I_{final} and I_{out} are both one because these processes are only executed when outputting the resultant data.

As a result, the throughput and latency can be expressed as

$$Th = M_p \times \frac{f_{max}}{\frac{M_p}{B}(I_{in} + I_{core}) + I_{final} + I_{out}}, \quad (2)$$

$$L = \frac{M_p}{Th} \ . \quad (3)$$

When M_p is sufficiently large, for example in the case of hashing a long message, I_{final} and I_{out} can be negligible from Eq. (2). In this case, the throughput $Th_{LongMessage}$ is approximated as

$$Th_{LongMessage} = \frac{Bf_{max}}{I_{in} + I_{core}}. \quad (4)$$

On the other hand, when M_p is small, for example in the case of hashing a short message for authentication, we cannot ignore I_{final} and I_{out}. Moreover, as latency is an important metric for a short message rather than throughput, we use Eq. (3) to compare the speed performance of the SHA-3 candidates.

Table I shows the evaluation metrics. Here, the throughput of the core function block Th_{core} is

$$Th_{core} = M_p \times \frac{f_{max}}{\frac{M_p}{B}I_{core} + I_{final}}. \quad (5)$$

TABLE I
Evaluation Metrics.

	Long Message (Throughput)	Short Message (Latency)
Interface + Core	$\frac{B \cdot f_{max}}{I_{in}+I_{core}}$	$\frac{M_p}{Th}$
Core Function Block	$\frac{B \cdot f_{max}}{I_{core}}$	$\frac{M_p}{Th_{core}}$

V. Implementation Results for All Second-round Candidates

In this work, we implement SHA-256 and all second-round SHA-3 candidates aiming at a high-speed hardware implementation. [1] Although it is not possible to completely factor

[1]We plan to release the Verilog/VHDL source code for these 15 algorithms at http://www.rcis.aist.go.jp/special/SASEBO/SHA3-en.html.

TABLE II
RESULTS OF THE SHA-3 CANDIDATES ON VIRTEX-5 (XC5VLX30-3FF324).

Implementation	Input Block Size [bits]	Max. Clock Freq [MHz]	Total Number of Clock Cycles [cycles] I/F + Core	Total Number of Clock Cycles [cycles] Core	Long Message Throughput [Mbps]	Short Message M=1,024 Latency [μs]	Number of Occupied Slices	Number of Slice Registers	Number of Slice LUTs	Power [W] Long Msg	Power [W] Short Msg	Energy [nJ/bit] Long Msg	Energy [nJ/bit] Short Msg
SHA-256	512	260	148 (196)	68 (68)	899 (1,958)	1.89 (0.785)	609	1,224	2,045	0.21	0.21	1.17	1.17
BLAKE-32	512	115	121 (169)	22 (22)	487 (2,676)	3.57 (0.574)	1,660	1,393	5,154	0.27	0.27	0.49	0.49
BMW-256	512	34	98 (148)	2 (4)	178 (8,704)	10.12 (0.235)	4,350	1,317	15,012	0.41	0.41	0.07	0.09
CubeHash16/32-256	256	185	64 (272)	16 (176)	740 (2,960)	2.85 (1.297)	590	1,316	2,182	0.23	0.23	0.61	1.82
ECHO-256	1,536	149	407 (455)	99 (99)	562 (2,312)	3.05 (0.664)	2,827	4,198	9,885	0.33	0.33	0.89	0.89
Fugue-256	32	78	8 (93)	2 (39)	312 (1,248)	4.47 (1.321)	4,013	1,043	13,255	0.36	0.37	0.95	1.47
Grøstl-256	512	154	106 (164)	10 (20)	744 (7,885)	2.44 (0.260)	2,616	1,570	10,088	0.31	0.31	0.25	0.25
Hamsi-256	32	210	10 (63)	4 (9)	672 (1,680)	1.92 (0.690)	718	841	2,499	0.23	0.23	1.19	1.27
JH-256	512	201	135 (183)	39 (39)	762 (2,639)	2.25 (0.582)	2,661	1,612	8,392	0.25	0.25	0.80	0.80
Keccak(-256)	1,024	205	217 (265)	25 (25)	967 (8,397)	2.35 (0.244)	1,433	2,666	4,806	0.29	0.29	0.29	0.29
Luffa-256	256	261	57 (114)	9 (18)	1,172 (7,424)	1.31 (0.207)	1,048	1,446	3,754	0.24	0.24	0.36	0.43
Shabal-256	512	228	143 (341)	50 (200)	816 (2,335)	2.75 (1.316)	1,251	2,061	4,219	0.23	0.23	1.19	2.15
SHAvite-3_{256}	512	251	134 (185)	38 (38)	959 (3,382)	1.79 (0.454)	1,063	1,363	3,564	0.24	0.24	0.73	0.73
SIMD-256	512	75	142 (190)	46 (46)	270 (835)	6.32 (1.840)	3,987	6,693	13,908	0.29	0.29	1.09	1.09
Skein-256-256	256	115	75 (143)	21 (41)	393 (1,402)	3.20 (0.904)	854	929	2,864	0.26	0.26	0.86	1.08

out the designer in our comparison, the 15 algorithms were all prototyped and tested using the same evaluation platform. Each was evaluated according to the metrics indicated above, covering performance, area, power consumption, and energy consumption.

Table II shows a comprehensive summary of the measurement results. As with all measurement data, it is important to understand the assumptions used when collecting these numbers. The table includes the following quantities for each candidate.

- Message block size
- Highest clock frequency achievable with the design on a Virtex-5 FPGA
- Latency in terms of clock cycles
- Throughput of the design in clock megabits per second and latency of the design for short messages, in microseconds. These numbers assume that the FPGA is operating at the maximum achievable clock frequency for the given design.
- Area cost of the design, in terms of occupied Virtex-5 slices, number of flip-flops, and number of LUTs.
- Power consumption of the design for long and short messages. The power consumption is measured directly on the core power supply (V_{vccint}) of the FPGA, with the FPGA operating at 24 MHz.
- Energy consumption of the design for long and short messages. The energy consumption is normalized against the block size, and expressed in nJ/bit.

VI. CONCLUSIONS

In this paper, we propose an evaluation platform and a consistent evaluation method to implement fair hardware evaluation of cryptographic hash functions. The platform includes a SASEBO-GII evaluation board, evaluation software, and appropriate interface definition. Using this method, we implement all second-round SHA-3 candidates and obtain the resulting cost and performance factors.

ACKNOWLEDGMENT

This effort was supported in part by NIST.

REFERENCES

[1] W. E. Burr, "Cryptographic Hash Standards: Where Do We Go from Here?," IEEE Security & Privacy, Vol. 4, no 2, p. 88-91, 2006.

[2] National Institute of Standards and Technology (NIST), "Cryptographic Hash Algorithm Competition," http://csrc.nist.gov/groups/ST/hash/sha-3/index.html.

[3] S. Tillich, M. Feldhofer, M. Kirschbaum, T. Plos, J. Schmidt, and A. Szekely, "High-Speed Hardware Implementations of BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein," IACR ePrint archive, 2009/510, 2009.

[4] A. H. Namin and M. A. Hasan, "Hardware Implementation of the Compression Function for Selected SHA-3 Candidates," CACR 2009-28, 2009.

[5] B. Baldwin, A. Byrne, M. Hamilton, N. Hanley, R. P. McEvoy, W. Pan, and W. P. Marnane, "FPGA Implementations of SHA-3 Candidates:CubeHash, Grøstl, LANE, Shabal and Spectral Hash," IACR ePrint Archive, Report 2009/342, 2009.

[6] B. Jungk, S. Reith, and J. Apfelbeck, "On Optimized FPGA Implementations of the SHA-3 Candidate Grøstl," IACR ePrint Archive, Report 2009/206, 2009.

[7] CERG at George Mason University. Hardware Interface of a Secure Hash Algorithm (SHA). Functional Specification, October 2009.

[8] B. Baldwin, A. Byrne, L. Lu, M. Hamilton, N. Hanley, M. O'Neill, and W. Marnane, "A Hardware Wrapper for the SHA-3 Hash Algorithms," IACR ePrint archive, 2010/124, 2010.

[9] "National Institute of Advanced Industrial Science and Technology (AIST), Research Center for Information Security (RCIS): "Side-channel Attack Standard Evaluation Board (SASEBO)"," http://www.rcis.aist.go.jp/special/SASEBO/SASEBO-GII-en.html.

[10] Z. Chen, S. Morozov, and P. Schaumont, "A Hardware Interface for Hashing Algorithms," IACR ePrint archive, 2008/529, 2008.

[11] K. Kobayashi, J. Ikegami, S. Matsuo, K. Sakiyama, and K. Ohta, "Evaluation of Hardware Performance for the SHA-3 Candidates Using SASEBO-GII," IACR ePrint archive, 2010/10, 2010.

[12] ECRYPT II, "SHA-3 Hardware Implementations," http://ehash.iaik.tugraz.at/wiki/SHA-3_Hardware_Implementations.